Week 2

Task2:

Lambda expressions

Aim:- To implement three methods in Java 8 using lambda expressions that check whether a number is:Odd or Even,Prime or Composite,Palindrome or Not.

Algorithm:-

1. Define Functional Interface

- Create Perform Operation with a single abstract method boolean check (int a).

2. Implement Lambdas

- Odd/Even: Return true if a % 2 != 0, else false.
- Prime:
  - If a <= 1, return false.
  - If a == 2, return true.
  - If divisible by 2, return false.
  - Loop from 3 to √a, check divisibility.
- Palindrome:
  - Convert number to string.
  - Reverse string and compare with original.

3. Checker Method

- Accepts a PerformOperation and a number, applies the lambda, and returns the result.

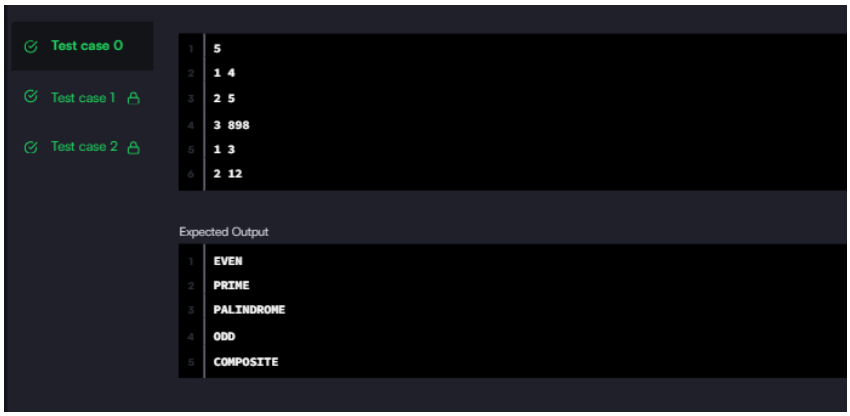
4. Main Program Flow

- Read number of test cases T.
- For each test case, read choice and number.
- Call the appropriate lambda method.
- Print result in required format (ODD/EVEN, PRIME/COMPOSITE, PALINDROME/NOT PALINDROME).

Program:-

```
public static PerformOperation isOdd() {
        return a -> a % 2 != 0;
    }

    public static PerformOperation isPrime() {
        return a -> {
            if (a <= 1) return false;
            for (int i = 2; i <= Math.sqrt(a); i++) {
                if (a % i == 0) return false;
            }
            return true;
        };
    }

    public static PerformOperation isPalindrome() {
        return a -> {
            int temp = a, rev = 0;
            while (temp > 0) {
                rev = rev * 10 + temp % 10;
                temp /= 10;
            }
            return rev == a;
        };
    }
}
```

Output:-



Result:- Thus, To implement three methods in Java 8 using lambda expressions that check whether a number is:Odd or Even,Prime or Composite,Palindrome or Not is are successfully executed.

Task3:

MIN-MAX PROBLEM

Aim:-Find the minimum and maximum sum of 4 out of 5 elements in the given array.

Algorithm:-

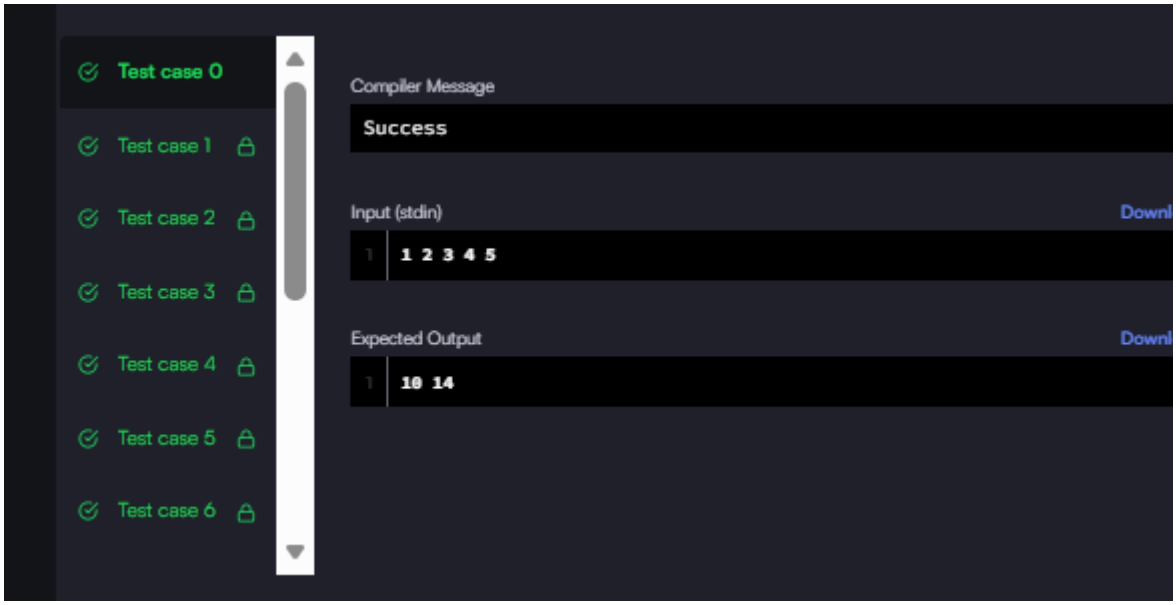
1. Initialize min and max with the first element of the array, and sum to 0.
2. Iterate through the array, updating min, max, and sum accordingly.
3. Calculate minSum and maxSum by subtracting max and min from sum, respectively.
4. Return minSum and maxSum.

Program:-

```
import java.io.*;
import java.util.*;

public class Solution {
    public static void miniMaxSum(int[] arr) {
        long total = 0;
        int min = arr[0], max = arr[0];

        for (int num : arr) {
            total += num;
            if (num < min) min = num;
            if (num > max) max = num;
        }

        long minSum = total - max;
```

```
        long maxSum = total - min;

        System.out.println(minSum + " " + maxSum);
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int[] arr = new int[5];
        for (int i = 0; i < 5; i++) {
            arr[i] = sc.nextInt();
        }
        miniMaxSum(arr);
        sc.close();
    }
}
```

Output:-



Result:-Thus,The minimum and maximum sum of 4 out of 5 elements in the given array is are successfully executed.

Task5:

Is Palindrome (string)

Aim:- To write a Java function that determines whether a given string is a palindrome (case-insensitive).Return 2 if the string is a palindrome,Return 1 if it is not a palindrome.

Algorithm:-

1. Input: A single word string (input1).
2. Convert the string to lowercase to ensure case-insensitive comparison.
3. Reverse the string using StringBuilder.reverse().
4. Compare the original lowercase string with the reversed string.

- If they are equal → return 2.
- Else → return 1.

Program:-

```
import java.io.*;
import java.util.*;


class UserMainCode {
   public int isPalindrome(String input1) {
      String str = input1.toLowerCase();


      String reversed = new StringBuilder(str).reverse().toString();
      if (str.equals(reversed)) {
         return 2;
      } else {
         return 1;
      }
   }
}
```

Output:-

```
PS D:\java> javac practice.java
PS D:\java> java practice
false
```

Result:-Thus, To write a Java function that determines whether a given string is a palindrome (case-insensitive).Return 2 if the string is a palindrome,Return 1 if it is not a palindrome is are successfully executed.

Task6:

All Digits Count

Aim:-To Count the total number of digits in a given positive integer using iterative division.

Algorithm:-

1. Initialize count = 0.
2. While num > 0:
   a. Increment count by 1.
   b. Update num = num / 10 (integer division).
3. Return count.

Program:-

```
class UserMainCode {
   public static int digitCount(int num) {

      int count = 0;

      while (num != 0) {
         count++;
         num = num / 10;
      }

      return count;
   }
```

}

Output:-

```
PS D:\java> javac practice.java
PS D:\java> java practice
4
```

Result:-Thus,To Count the total number of digits in a given positive integer using iterative division is are successfully executed.

Task7:

JAVA DATE & TIME

Aim:- The aim of this problem is to determine the day of the week (e.g., MONDAY, TUESDAY, etc.) for a given date specified by month, day, and year.

Algorithm:-

1. Input: Read three integers → month, day, year.
2. Initialize Calendar:
    - Use Calendar.getInstance().
    - Set the date with cal.set(year, month - 1, day) (since Calendar months are 0-based).
3. Find Day of Week:
    - Retrieve the day name using cal.getDisplayName(Calendar.DAY_OF_WEEK, Calendar.LONG, Locale.ENGLISH).
4. Format Output:
    - Convert the result to uppercase.
5. Return: Print the day of the week.

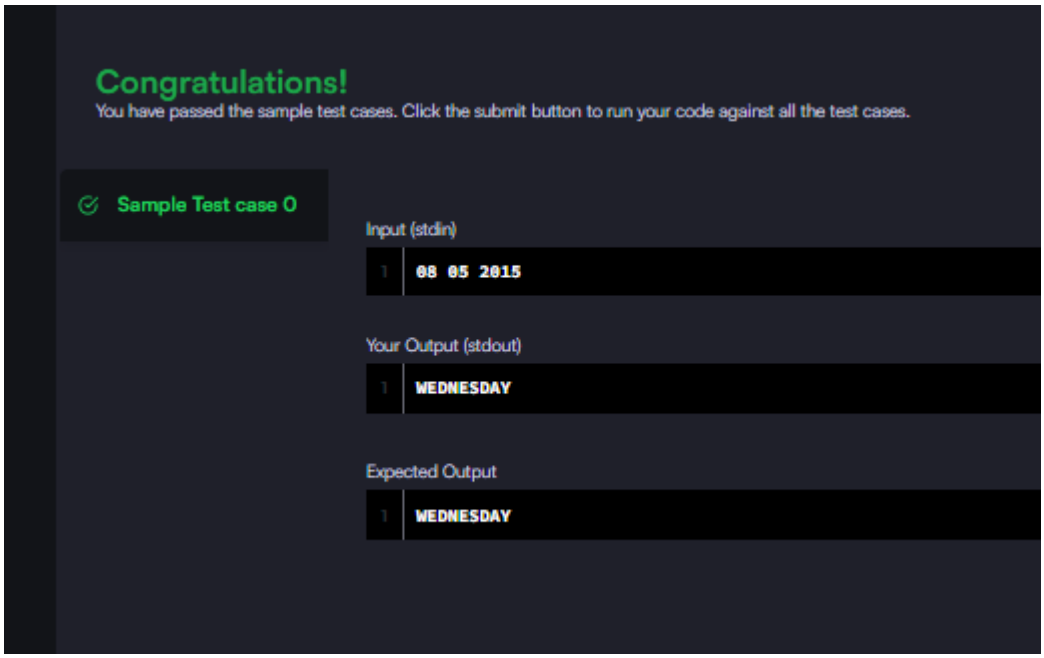
Program:-

```
import java.io.*;

import java.math.*;

import java.security.*;

import java.text.*;

import java.util.*;

import java.util.concurrent.*;

import java.util.function.*;

import java.util.regex.*;
```

```
import java.util.stream.*;
import static java.util.stream.Collectors.joining;
import static java.util.stream.Collectors.toList;
import java.util.*;

class Result {
    public static String findDay(int month, int day, int year) {
        Calendar cal = Calendar.getInstance();
        cal.set(year, month - 1, day);
        String dayOfWeek = cal.getDisplayName(Calendar.DAY_OF_WEEK,
Calendar.LONG, Locale.ENGLISH);
        return dayOfWeek.toUpperCase();
    }
}

public class Solution {
    public static void main(String[] args) throws IOException {
        BufferedReader bufferedReader = new BufferedReader(new
InputStreamReader(System.in));
        BufferedWriter bufferedWriter = new BufferedWriter(new
FileWriter(System.getenv("OUTPUT_PATH")));

        String[] firstMultipleInput = bufferedReader.readLine().replaceAll("\\s+$",
"").split(" ");

        int month = Integer.parseInt(firstMultipleInput[0]);

        int day = Integer.parseInt(firstMultipleInput[1]);
```

```
        int year = Integer.parseInt(firstMultipleInput[2]);

        String res = Result.findDay(month, day, year);

        bufferedWriter.write(res);
        bufferedWriter.newLine();

        bufferedReader.close();
        bufferedWriter.close();
    }
}
```

Output:-



Result:-Thus,To this problem is to determine the day of the week (e.g., MONDAY, TUESDAY, etc.) for a given date specified by month, day, and year is are successfully executed.

Task8:

Weight of a hill pattern

Aim:- To calculate the total weight of a hill pattern given:input1: total levels in the hill (rows),input2: weight of the head level (first row),input3: weight increment for each subsequent level.

Algorithm:-

1. Initialize total weight = 0.
2. Loop through each level i from 1 to input1:
    - Compute the weight of the current level:
      levelWeight = input2 + (i - 1) * input3
    - Compute the number of stars in this level: stars = i
    - Add to total: totalWeight += levelWeight * stars
3. Return totalWeight.

Program:-

```
import java.io.*;
import java.util.*;


class UserMainCode {
   public int totalHillWeight(int input1, int input2, int input3) {
      int totalWeight = 0;


      for (int i = 1; i <= input1; i++) {
         int levelWeight = input2 + (i - 1) * input3;
         totalWeight += levelWeight * i;
      }
```

```
        return totalWeight;
    }
}
```

Output:-

```
PS D:\java> javac practice.java
PS D:\java> java practice
90
```

Result:- Thus,To calculate the total weight of a hill pattern given:input1: total levels in the hill (rows),input2: weight of the head level (first row),input3: weight increment for each subsequent level is are successfully executed.

Task10:

Sum of Sums of Digits in Cyclic order

Aim:- To compute the sum of sums of digits of a given number in cyclic order. That means:Start from the last digit, keep adding digits cumulatively (suffix sums),Add each cumulative sum to the final answer.

Algorithm:-

1. Convert the number into a string to easily access digits.
2. Initialize suffixSum = 0 and answer = 0.
3. Traverse digits from **right to left** (last digit to first).
    - Add the current digit to suffixSum.
    - Add suffixSum to answer.
4. Return answer.

Program:-

```
import java.io.*;
import java.util.*;
class UserMainCode
{
public int sumOfSumsOfDigits (int input1){
String s = String.valueOf(input1);
int n = s.length();
int answer = 0;
int suffixSum = 0;
for (int i = n-1; i >=0; i--){
suffixSum += s.charAt(i) - '0';
```

```
answer += suffixSum;
}
return answer;
}
}
```

Output:-

```
PS D:\java> javac practice.java
PS D:\java> java practice
40
```

Result:-Thus, To compute the sum of sums of digits of a given number in cyclic order.That means:Start from the last digit, keep adding digits cumulatively (suffix sums),Add each cumulative sum to the final answer is are successfully executed.